INFORMATION AND CONTROL 42, 290–304 (1979)

Maximal Derivations for Probabilistic Strings in Stochastic Languages*

SHMUEL PELEG

Computer Science Center, University of Maryland, College Park, Maryland 20742

A probabilistic string is a sequence of probability vectors. Each vector specifies a probability distribution over the possible symbols at its location in the string. In a probabilistic grammar a probability is assigned to every derivation. Given a probabilistic string and a probabilistic grammar the concept of a maximal derivation is defined. Algorithms for finding the maximal derivation for probabilistic finite state and linear grammars are given. The case where a waveform can be segmented into several possible probabilistic strings is also considered.

1. INTRODUCTION

A probabilistic grammar assigns a probability to every derivation in the grammar. Let G be a probabilistic grammar having T as its set of terminals, and let D be a derivation (generating the string x_D). We denote the probability assigned to the derivation D in the grammar G by $P_G(D)$.

Given a string $V = v_1 \cdots v_L$ of probability vectors $v_i : T \to [0, 1]$, the probability of any given string $x = x_1 \cdots x_L$ being an interpretation of V is defined by $P_V(x) = \prod_{i=1}^{L} v_i(x_i)$.

In order for D to be a derivation that produces an interpretation for V the following two events must occur:

(a) D is generated by the grammar G (regardless of V).

(b) x_D is an interpretation for V (regardless of G).

Since (a) and (b) are independent, the probability that both will occur is the product of their individual probabilities:

$$P(D; V) \equiv \text{Prob}(D \text{ generates an interpretation for } V) = P_G(D) \cdot P_V(x_D) \quad (1)$$

Given a probabilistic grammar G and a probabilistic string V, it is of interest to find *maximal derivations* D, i.e., derivations that maximize $P(D; V)$. There

* The support of the National Science Foundation under Grant MCS-76-23763 is gratefully acknowledged, as is the help of Ms. Kathryn Riley in preparing this paper. The author also wishes to thank A. Rosenfeld, R. Kirby, and R. Hamlet for many helpful discussions.

0019-9958/79/090290–15$02.00/0

are several reasons one might want to find such derivations. When the probabilistic string is the response of a detector to the output of a stochastic process that can be modeled by a probabilistic grammar, the maximal derivation is the best candidate for the sequence of states of the stochastic process. A similar approach that considers heart pulses as a stochastic process is used by Albus (1977). Even when derivations themselves are of no interest, and the grammar is used only to assign probabilities to strings, a maximal derivation defines a string that can be regarded as an optimal disambiguation of the probabilistic string.

A more general problem is when a waveform can be segmented into a sequence of segments in many possible ways. For such cases of ambiguous segmentation a maximal derivation will also be defined.

The general problem of finding maximal derivations is very complicated, since all derivations of all strings of length L must be tested, and the number of these derivations is usually exponential in L. In order to find maximal derivations we need to use efficient search methods such as the A^* algorithm (Nilsson, 1971) that uses heuristic knowledge to guide the search process. In this paper, rather than using heuristic search for general grammars, simple classes of grammars are considered. Maximal derivations are found in linear time for languages generated by probabilistic finite state grammars, and in quadratic time for probabilistic linear grammars. In the case of ambiguous segmentation, a maximal derivation is found for finite state grammars in time linear in the number of possible segments.

2. Probabilistic Grammars

In this section probabilistic grammars for generating stochastic languages are defined. We use the definition of Salomaa (1969), rather than the stochastic grammar definition given by Fu (1974). This was done because Fu's stochastic grammars can be immediately translated into probabilistic grammars with the same productions, whereas translation of probabilistic grammars into Fu's stochastic grammars involves expansion of the number of productions. Another consideration is that probabilistic grammars are more powerful in the context free case.

Definition. A context free probabilistic grammar is a 4-tuple $G_P = (N, T, P, S)$:

N	Finite set of nonterminals
T	Finite set of terminals
$S \in N$	Start symbol

P Finite set of productions of the form i: (A, α, φ_i), where $i \geqslant 1$ is an index, $A \in N$, $\alpha \in (N \cup T)^*$, and $\varphi_i \in [0, 1]^{|P|}$, $|P|$ being the number of productions. $\varphi_i(j)$, the jth term of φ_i, is the probability of applying production j after production i has been applied.

A production can also be written as

$$[i\colon A \to \alpha;\ \varphi_i = (p_1, \ldots, p_{|P|})].$$

Here every φ_i is a probability vector, and $\sum_j \varphi_i(j) = 1$ for all $i > 0$. We also define φ_0 as the initial probability distribution over the productions for the start symbol S.

A string $\xi A\eta$, $A \in N$, $\xi, \eta \in (N \cup T)^*$ can be rewritten as $\xi\alpha\eta$ if a production n: (A, α, φ_n) is in P. This is denoted by $\xi A\eta \Rightarrow_n \xi\alpha\eta$. Any derivation $D \equiv S \Rightarrow_{i_1} \cdots \Rightarrow_{i_L} x_D$, $x_D \in T^*$, has probability

$$P(D) = \varphi_0(i_1) \cdot \prod_{k=1}^{L-1} \varphi_{i_k}(i_{k+1}). \tag{2}$$

The following is a simple example of a finite state probabilistic grammar:

		$\varphi_0 = (\frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0)$
1:	$S \to aS_1$	$\varphi_1 = (0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0)$
2:	$S \to bS_2$	$\varphi_2 = (0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})$
3:	$S_1 \to aS_1$	$\varphi_3 = (0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0)$
4:	$S_1 \to a$	$\varphi_4 = (0, 0, 0, 1, 0, 0)$
5:	$S_2 \to bS_2$	$\varphi_5 = (0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})$
6:	$S_2 \to b$	$\varphi_6 = (0, 0, 0, 0, 0, 1)$

This grammar generates the language $\{a^n \mid n > 1\} \cup \{b^n \mid n > 1\}$, in which every sentence of length $k > 1$ has the probability $(\frac{1}{2})^k$.

In the next section we discuss the choice of one (maximal) derivation out of all the possible derivations in a probabilistic grammar for a given string.

3. Maximal Derivations of Strings in Finite-State Grammars

Given a sentence $x = x_D$, a derivation D of x_D with the highest probability is one that maximizes $P(D)$ as defined in (2). To see this, let $P(x_D)$ be the probability that x_D is generated by the grammar, independently of whether or not D was used. Then

$$P(D \mid x_D) = \frac{P(x_D \mid D) \cdot P(D)}{P(x_D)} = \frac{P(D)}{P(x_D)}$$

since $P(x_D \mid D) = 1$. Thus $P(D \mid x_D)$ is maximized when $P(D)$ is. We call a derivation that maximizes $P(D)$ a maximal derivation for x.

Finite-state probabilistic languages are those generated by context-free probabilistic grammars whose productions are of the form $[i: A \to xB; \varphi_i]$, $x \in T$, $B \in N \cup \{\lambda\}$, where λ is the empty symbol. A maximal derivation in a finite state probabilistic language can be found by a method similar to forward dynamic programming, as is done in the Viterbi algorithm (Forney, 1973) for Markov processes, and as described in Albus (1977) for finite-state probabilistic automata. Since for a finite-state probabilistic grammar the state of a derivation at a given point is dependent only on the nonterminal and the previous production, it can be represented by a pair $\langle X, i\rangle$, $i \geqslant 0$, $X \in N \cup \{\#\}$, where $\#$ stands for the end of the string. $\langle X, i\rangle$ is the state in which the current nonterminal is X, and the next production is chosen according to the probability vector φ_i (i was the last production applied.)

The algorithm scans the input sentence from left to right, finding the maximal probability for every state at every step. When the end of the sentence is reached, we find a terminal state (a state with "#") that has maximum probability, and trace the sequence of productions that lead to that state.

Formally, given the input sentence $x = x_1 \cdots x_L$, define $P^{(k)}(\langle X, i\rangle)$ to be the maximal probability of reaching the state $\langle X, i\rangle$ at the kth position. Initially, all $P^{(0)}(\langle X, i\rangle) = 0$, except for $P^{(0)}(\langle S, 0\rangle) = 1$. Then we can define

$$P^{(k)}(\langle X, i\rangle) = \operatorname*{Max}_{Y,j}\{P^{(k-1)}(\langle Y, j\rangle) \cdot \varphi_j(i) \mid \text{Production } i \text{ is } Y \to x_k X\},$$
and (3)
$$P^{(k)}(\langle \#, i\rangle) = \operatorname*{Max}_{Y,j}\{P^{(k-1)}(\langle Y, j\rangle) \cdot \varphi_j(i) \mid \text{Production } i \text{ is } Y \to x_k\}.$$

Let $F^{(k)}(\langle X, i\rangle)$ be a state $\langle Y, j\rangle$ that maximizes $P^{(k)}(\langle X, i\rangle)$. Let $(\langle X_k, i_k\rangle)_{k=1}^{L}$ be a sequence of states such that $\langle X_{k-1}, i_{k-1}\rangle = F^{(k)}(\langle X_k, i_k\rangle)$, and $\langle X_L, i_L\rangle = \langle \#, i_L\rangle$ is a terminal state with maximum probability in the Lth position. Such a sequence can be easily formed by tracing back from $\langle \#, i_L\rangle$ using the F's. It can be seen that $i_1, \ldots, i_L$ of that sequence is a maximal derivation for the input x, and is found in linear time.

EXAMPLE 1. Let G_1 be the grammar having the following productions:

		$\varphi_0 = (1, 0, 0, 0, 0)$
1:	$S \to aA$	$\varphi_1 = (0, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2})$
2:	$A \to aA$	$\varphi_2 = (0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$
3:	$A \to aS$	$\varphi_3 = (1, 0, 0, 0, 0)$
4:	$A \to a$	$\varphi_4 = (0, 0, 0, 1, 0)$
5:	$A \to bA$	$\varphi_5 = (0, \frac{1}{2}, \frac{1}{2}, 0, 0)$

and let *aaaaabaa* be the string to be parsed.

Figure 1 is the parsing table for this example. The kth column in this table indicates the values of $P^{(k-1)}(\langle X, i\rangle)$ for all possible pairs of $\langle X, i\rangle$. The links connect $P^{(k)}(\langle X, i\rangle)$ to a $P^{(k-1)}(\langle Y, j\rangle)$ that maximizes their probability, and are essentially the $F^{(k)}(\langle X, i\rangle)$ of the algorithm. From the table, a maximal derivation for *aaaaabaa* is $S \Rightarrow_1 aA \Rightarrow_2 aaA \Rightarrow_2 aaaA \Rightarrow_3 aaaaS \Rightarrow_1 aaaaaA \Rightarrow_5 aaaaabA \Rightarrow_2 aaaaabaA \Rightarrow_4 aaaaabaa$ and has probability $(\frac{1}{2})^{10}$. It is obtained by tracing back from the only terminal state at the last position of the string, $\langle \#, 4\rangle$, and indicated by thicker links in Fig. 1.

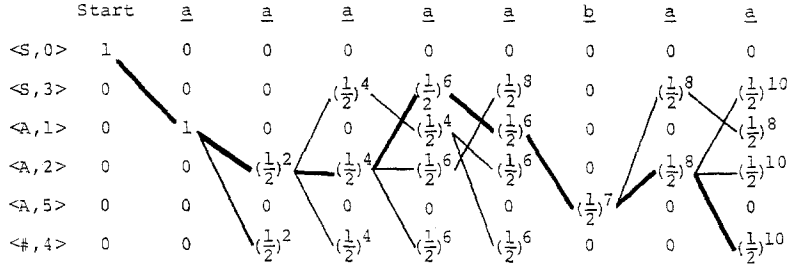

FIG. 1. Parsing table for Example 1.

4. MAXIMAL DERIVATIONS OF PROBABILISTIC STRINGS IN FINITE-STATE GRAMMARS

An important property of the algorithm described in Section 3 is that it can be naturally generalized to probabilistic sentences. A probabilistic sentence is a sequence $V = v_1 \cdots v_L$ of vectors v_i: $T \to [0, 1]$, where $v_i(t)$ is the probability that the ith symbol is the terminal t. We can regard a probabilistic sentence as the output of a local detector that cannot decide on the exact symbol in that location, and gives a probability vector over the possible interpretations. This is very reasonable in cases such as speech analysis and character recognition, where the detector cannot determine the exact interpretation based on local evidence only.

Given a probabilistic string $V = v_1 \cdots v_L$ and a finite-state probabilistic grammar G_p, the probability that the derivation $D \equiv S \Rightarrow_{i_1} \cdots \Rightarrow_{i_L} x_D$, $x_D = x_1 \cdots x_L \in T^*$, will be a derivation for an interpretation of V is given, using (1) and (2), by

$$P(D; V) = P_G(D) \cdot P_V(x_D)$$

$$= \varphi_0(i_1) \cdot \prod_{k=1}^{L-1} \varphi_{i_k}(i_{k+1}) \cdot \prod_{k=1}^{L} v_k(x_k).$$

This can be rewritten as

$$P(D; V) = \varphi_0(i_1) \cdot \varphi_1(t_{i_1}) \cdot \prod_{k=1}^{L-1} [\varphi_{i_k}(i_{k+1}) \cdot v_{k+1}(t_{i_{k+1}})], \tag{4}$$

where t_{i_n} is the terminal written by the i_nth production, and is also the nth symbol in the string generated by D. When V is a sequence of unit vectors, (4) reduces to (2).

A maximal derivation is found by an algorithm similar to the one in Section 3. Initially, all $P^{(0)}(\langle X, i\rangle) = 0$, except $P^{(0)}(\langle S, 0\rangle) = 1$. Then, as in Section 3 we define

$$P^{(k)}(\langle X, i\rangle) = \max_{Y,j}\{P^{(k-1)}(\langle Y, j\rangle) \cdot \varphi_j(i) \cdot v_k(t_i) | \text{ Production } i \text{ is } Y \to t_i X\} \tag{5}$$

and

$$P^{(k)}(\langle \#, i\rangle) = \max_{Y,j}\{P^{(k-1)}(\langle Y, j\rangle) \cdot \varphi_j(i) \cdot v_k(t_i) \mid \text{Production } i \text{ is } Y \to t_i\}.$$

Again, let $F^{(k)}(\langle X, i\rangle)$ be a state $\langle Y, j\rangle$ that maximizes $P^{(k)}(\langle X, i\rangle)$. Definition (5) reduces to definition (3) when V is composed of unit vectors only. A maximal derivation is found exactly as in Section 3.

EXAMPLE 2. As an illustration of this algorithm we use the grammar G_1 of Example 1. Since the terminals for G_1 are $\{a, b\}$, V will be a sequence of pairs $\binom{p_1}{p_2}$, where p_1 is the probability of a and p_2 is the probability of b. Figure 2 illustrates how the algorithm works for sequences of pairs having $p_1 = p_2 = 0.5$, which are the most ambiguous sequences in a two-terminal language. From the table we see that the unique maximal derivation for $\binom{1/2}{1/2}\binom{1/2}{1/2}$ is $S \Rightarrow_1 aA \Rightarrow_4 aa$.

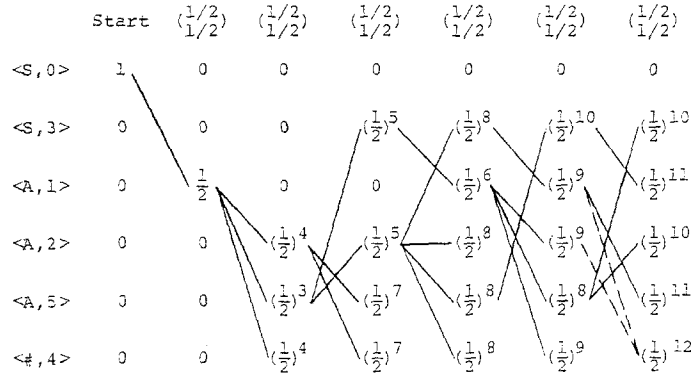

FIG. 2. Parsing table for Example 2.

For $(\frac{1/2}{1/2})^3$ it is $S \Rightarrow_1 aA \Rightarrow_2 aaA \Rightarrow_4 aaa$.

For $(\frac{1/2}{1/2})^4$ it is $S \Rightarrow_1 aA \Rightarrow_5 abA \Rightarrow_2 abaA \Rightarrow_4 abaa$.

For $(\frac{1/2}{1/2})^5$ it is $S \Rightarrow_1 aA \Rightarrow_5 abA \Rightarrow_3 abaS \Rightarrow_1 abaaA \Rightarrow_4 abaa$.

For $(\frac{1/2}{1/2})^6$ there are two maximal derivations:

1. $S \Rightarrow_1 aA \Rightarrow_5 abA \Rightarrow_3 abaS \Rightarrow_1 abaaA \Rightarrow_2 abaaaA \Rightarrow_4 abaaaa$.
2. $S \Rightarrow_1 aA \Rightarrow_5 abA \Rightarrow_2 abaA \Rightarrow_3 abaaS \Rightarrow_1 abaaaA \Rightarrow_4 abaaaa$.

These maximal derivations were obtained by tracing back from the only terminal state, $\langle \#, 4 \rangle$, at the corresponding position. Example 2 shows how a maximal derivation can be found for a probabilistic string in a finite state probabilistic grammar even if the string itself is very ambiguous. It is also seen that the "future" of the string influences the choice of the "past". When the string is of length 2 or 3, "a" is the second terminal of the maximal interpretation, while for longer strings "b" is the second terminal of the maximal interpretation.

5. Maximal Derivations of Probabilistic Strings in Linear Grammars

In a linear grammar $G_L = (N, T, P, S)$, each production has the form $A \to \alpha B \beta$, $A \in N$, $B \in N \cup \{\lambda\}$, and $\alpha, \beta \in T^*$. Linear grammars that do not generate the empty string can be rewritten in a normal form, where productions have the form $A \to xB$ or $A \to Bx$, $A \in N$, $B \in N \cup \{\lambda\}$ and $x \in T$ (Problem 2.4.32 in Aho and Ullman (1972)). In this section only grammars in normal form are considered.

Linear grammars are more powerful than finite-state grammars, but weaker than context free. Finding a maximal derivation for a linear grammar is more complicated than in the case of a finite-state grammar. When parsing a string $x_i \cdots x_j$ in a finite-state grammar, we know that x_i will be "consumed" for any production that can be applied, and the next step will be to parse $x_{i+1} \cdots x_j$. When parsing a string $x_i \cdots x_j$ in a linear grammar, either x_i is consumed and we continue with $x_{i+1} \cdots x_j$, or x_j is consumed and we continue with $x_i \cdots x_{j-1}$. Having the consider two possibilities at every step seems to lead to exponential complexity. However, many of these cases coincide. For example, $x_i \cdots x_j$ can be reached from $x_{i-1} \cdots x_j$ by consuming x_{i-1} or from $x_i \cdots x_{j+1}$ by consuming x_{j+1}. All the possible positions can be arranged in the structure displayed in Fig. 3. In this figure, a position in a certain level can be reached from either of its two parents in the higher level by one of the two ways described.

Thus, instead of having a sequence of $P^{(k)}$'s as in (5), we have an array $P^{(k,j)}$. Here $P^{(k,j)}(\langle X, i \rangle)$ is the probability of getting to state $\langle X, i \rangle$ with the current string $x_{k+1} \cdots x_{L-j}$, given $x = x_1 \cdots x_L$ as initial string. Given a grammar G_S

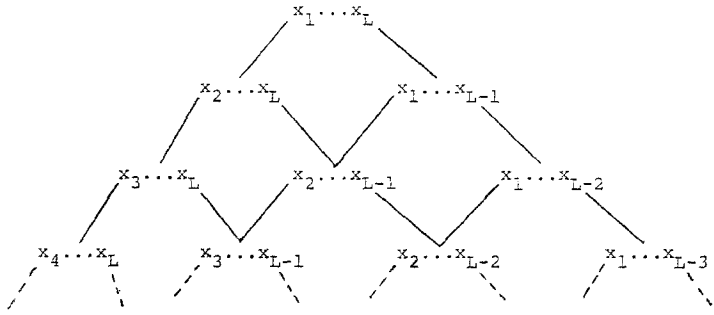

FIG. 3. Possible positions when parsing in linear grammar.

and a probabilistic sentence V the P's can be defined by a recursive formula similar to (5):

$$P^{(0,0)}(\langle X, i\rangle) = 0, \text{ except that } P^{(0,0)}(\langle S, 0\rangle) = 1.$$

and for $0 \leqslant n + m \leqslant L$:

$$\begin{aligned} P^{(m,n)}(\langle X, i\rangle) &= \operatorname{Max}\{\operatorname*{Max}_{Y,j}\{P^{(m-1,n)}(\langle Y, j\rangle) \cdot \varphi_j(i) \cdot v_m(t_i) \mid \text{Production } i \text{ is } Y \to t_i X\}, \quad (6) \\ &\qquad \operatorname*{Max}_{Y,j}\{P^{(m,n-1)}(\langle Y, j\rangle) \cdot \varphi_j(i) \cdot v_{L-n+1}(t_i) \mid \text{Production } i \text{ is } Y \to X t_i\}\}. \end{aligned}$$

In (6), $P^{(m,n)}(\langle X, i\rangle)$ is maximized over the two possible ways of getting to $x_{m+1} \cdots x_{L-n}$; from $x_m \cdots x_{L-n}$ by "consuming" x_m, or from $x_{m+1} \cdots x_{L-n+1}$ by "consuming" x_{L-n+1}. All the $P^{(m,n)}$'s that are outside $0 \leqslant n \leqslant m$ are considered as having value zero. They occur when computing $P^{(0,k)}$'s using $P^{(-1,k)}$'s or $P^{(k,0)}$'s using $P^{(k,-1)}$'s. $P^{(m,n)}(\langle \#, i\rangle)$, $m = L - n$, are defined similarly to (6), but with terminal productions.

Again, $F^{(m,n)}(\langle X, i\rangle)$ is an element $(i, j, \langle Y, k\rangle)$ which specifies that $P^{(m,n)}(\langle X, i\rangle)$ was maximized by $P^{(i,j)}(\langle Y, k\rangle)$.

A maximal derivation is built by tracing back from a maximal terminal state, and building a sequence of states according to F. Let $S = \{(m_k, n_k, \langle X_k, i_k\rangle)\}_{k=1}^{L}$ be a sequence of elements such that

(a) $\langle X_L, i_L\rangle = \langle \#, i_L\rangle$ is a state that maximizes the value of $P^{(n,L-n)}$, $L \geqslant n \geqslant 0$ over all productions i; let $P^{(m_L,n_L)}(\langle \#, i_L\rangle)$ be that maximal value.

(b) $(m_{k-1}, n_{k-1}, \langle X_{k-1}, i_{k-1}\rangle) = F^{(m_k,n_k)}(\langle X_k, i_k\rangle)$.

A maximal derivation is then

$$S \Rightarrow_{i_1} \cdots \Rightarrow_{i_L} x = x_1 \cdots x_L,$$

where $i_1 \cdots i_L$ are taken from the sequence S. It can be shown that $i_1 \cdots i_L$ is a maximal derivation for $V = v_i \cdots v_L$ given the grammar G_S, it can be found in time complexity which is proportional to the square of the length of the input.

EXAMPLE 3. As an example, we consider the grammar G_2:

		$\varphi_0 = (1, 0, 0, 0)$
1:	$S \rightarrow aS$	$\varphi_1 = (0, 1, 0, 0)$
2:	$S \rightarrow Sb$	$\varphi_2 = (\frac{1}{2}, 0, \frac{1}{4}, \frac{1}{4})$
3:	$S \rightarrow cS$	$\varphi_3 = (0, 0, \frac{1}{2}, \frac{1}{2})$
4:	$S \rightarrow c$	$\varphi_4 = (0, 0, 0, 1)$

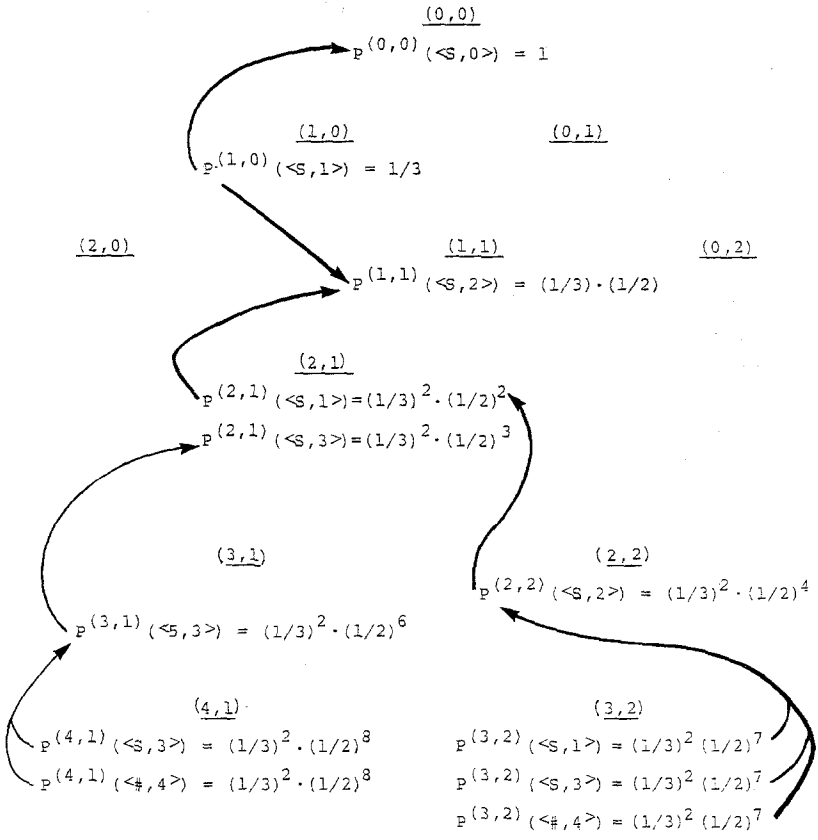

FIG. 4. Parsing network for Example 3.

$L(G_2)$ is the set $\{a^n c^k b^n \mid n, k \geqslant 1\}$ and $P(a^n c^k b^n) = (\frac{1}{2})^{n-1} \cdot \frac{1}{4} \cdot (\frac{1}{2})^{k-1}$. As a probabilistic string we will use

$$V = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \begin{pmatrix} 1/4 \\ 1/4 \\ 1/2 \end{pmatrix} \begin{pmatrix} 1/2 \\ 1/4 \\ 1/4 \end{pmatrix} \begin{pmatrix} 1/2 \\ 1/2 \\ 1/4 \end{pmatrix}.$$

In finding the maximal derivations, a network of positions (i, j) is used, and is displayed in Fig. 4. Under each position (i, j), there is a list of states that are possible when $v_{i+1} \cdots v_{5-j}$ is to be parsed, and the values of $P^{(i,j)}$ for these states. Every state is linked back to the state pointed to by the $F^{(i,j)}$. Except for the first two levels, positions that do not have nonzero states are not shown.

Tracing back from the maximal terminal state, we find that the maximal derivation used the following productions in reverse: 4, 2, 1, 2, 1. So, the maximal derivation is $S \Rightarrow_1 aS \Rightarrow_2 aSb \Rightarrow_1 aaSb \Rightarrow_2 aaSbb \Rightarrow_4 aacbb$.

6. Ambiguous Segmentation

A probabilistic string is too simple a representation for many problems, since using a probabilistic string assumes that a waveform can be segmented into individual parts (or letters). Often, however, the segmentation of the waveform is not obvious; many segmentations are possible, and the segmentation (e.g., into letters) should be chosen at the same time as the interpretation of every letter. As an example in the handwriting domain, the form *cl* can be interpreted either as the one letter "d", or the sequence of two letters "cl," depending on whether the segmentation chosen was into one letter or two letters. This ambiguity in segmentation cannot be represented by a probabilistic string. In this section a representation for strings with ambiguous segmentation is presented based on the work of Velasco and Rosenfeld (1978).

Given a waveform w in the interval (T_B, T_E), a segment $s_i = [b_i, e_i]$ is a time interval such that $T_B \leqslant b_i < e_i \leqslant T_E$. A segment $s_i = [b_i, e_i]$ is an *initial segment* if $b_i = T_B$, and is a *final segment* if $e_i = T_E$. Two segments $s_1 = [b_1, e_1]$ and $s_2 = [b_2, e_2]$ are *consecutive* if $e_1 = b_2$. In such a case s_2 is a *successor* of s_1, and s_1 is a *predecessor* of s_2.

A configuration $C = \{s_1, \ldots, s_n\}$, $s_i = [b_i, e_i]$, is called *complete* if

1. There exists a segment $s_i \in C$ such that s_i is an initial segment.
2. There exists a segment $s_j \in C$ such that s_j is a final segment.
3. For every segment $s_k \in C$ the following conditions hold:
   (a) $b_k = T_B$, or s_k has a predecessor in C.
   (b) $e_k = T_E$, or s_k has a successor in C.

In a complete configuration, each noninitial segment has at least one predecessor, and each nonfinal segment has at least one successor.

EXAMPLE 4. Let the waveform w be defined on the time interval [0, 10]. Consider the following segments:

$s_1 = [0, 10]$ (initial segment, final segment)
$s_2 = [0, 5]$ (initial segment)
$s_3 = [0, 3]$ (initial segment)
$s_4 = [3, 8]$
$s_5 = [5, 10]$ (final segment)
$s_6 = [8, 10]$ (final segment)
$s_7 = [2, 5]$

The configurations $C_0 = \{s_1\}$, $C_1 = \{s_1 , s_2 , s_5\}$ and $C_4 = \{s_1 , s_2 , s_3 , s_4 , s_5 , s_6\}$ are complete. The configuration $C_5 = \{s_2\}$ is not complete since it has no final segment, and the configuration $C_6 = \{s_2 , s_5 , s_7\}$ is not complete since neither s_2 nor s_5 is a predecessor of s_7. It is easy to show that for a complete configuration C for a waveform w, the time intervals of the segments in C cover the time interval for w.

A *segmentation* $S = \langle s_{i_1} ,..., s_{i_n} \rangle$ for a configuration C is a sequence of segments $s_i = [b_i , e_i]$ from C such that $b_{i_1} = T_B$, $e_{i_n} = T_E$, and for every $1 < i \leqslant n$ s_i is a successor of s_{i-1}. The sequences $\langle s_1 \rangle$, $\langle s_2 , s_5 \rangle$, $\langle s_3 , s_4, s_6 \rangle$ are all segmentations for the configuration C_4 of Example 4.

Let $\Lambda = \{t_1 , t_2 ,..., t_n\}$ be the set of possible labels (terminals) that can be assigned to every segment s_i of a complete configuration C. A *probabilistic configuration* PC is a complete configuration, where every segment $s \in PC$ has a probability vector P_s: $\Lambda \to [0, 1]$ associated with it. $P_s(t_i)$ represents the probability that segment s will be labeled by the label t_i.

A probabilistic configuration PC can induce a measure on every interpretation. Given a segmentation $S = \langle s_{i_1} ,..., s_{i_n} \rangle$ for PC, and an interpretation $T = t_{i_1} ,..., t_{i_n}$ for every segment, the measure P_S is defined by

$$P_S(T) = P(T \mid S) = \prod_{k=1}^{n} P_{s_{i_k}}(t_{i_k}). \tag{7}$$

EXAMPLE 5. Let the segments $s_1 ,..., s_7$ of Example 4 have the following probability vectors for the terminals {0, 1}:

$P_1(1) = 0$ $P_1(0) = 1$
$P_2(1) = 1$ $P_2(0) = 0$
$P_3(1) = 0.5$ $P_3(0) = 0.5$
$P_4(1) = 0$ $P_4(0) = 1$
$P_5(1) = 0$ $P_5(0) = 1$
$P_6(1) = 1$ $P_6(0) = 0$
$P_7(1) = 0.5$ $P_7(0) = 0.5$

Then for the segmentation $S = \langle s_1 \rangle$ we have $P_S(\text{'1'}) = 0$ and $P_S(\text{'0'}) = 1$. For the segmentation $S = \langle s_3 , s_4 , s_6 \rangle$, $P_S(\text{'111'}) = 0$, $P_S(\text{'000'}) = 0$, $P_S(\text{'101'}) = 0.5$, etc. Now that we have established a representation for a probabilistic string with ambiguous augmentation, maximal derivations can be found.

7. Maximal Derivations for Ambiguous Segmentation

As in Section 4, we have a probabilistic finite-state grammar $G_p = (N, T, P, S)$ with a probability $P_G(D)$ assigned to every derivation D in the grammar by (2). Let D be a derivation that produces the string $T = t_{i_1} ,..., t_{i_n}$, and let $S = \langle s_{i_1} ,..., s_{i_n} \rangle$ be a segmentation. We define a measure M_S for D as follows

$$\begin{aligned} M_S(D) &= P_G(D) \cdot P_S(T) \\ &= P_G(D) \cdot \prod_{k=1}^{n} P_{s_{i_k}}(t_{i_k}). \end{aligned} \tag{8}$$

A maximal derivation and segmentation pair consists of a derivation $D_{\max}$ and a segmentation $S_{\max}$ such that $M_{S_{\max}}(D_{\max})$ is the maximal value of $M_S(D)$ for all segmentations S and derivations D. The interpretation $T_{\max}$ generated by $D_{\max}$ will be a maximal unambiguous interpretation for the probabilistic configuration PC over the grammar G_p.

When the probabilistic grammar G_p is finite-state, an algorithm similar to that in Section 4 can be applied in order to find a maximal derivation. This algorithm is applied on a complete probabilistic configuration C represented as a graph (T, C), where the set of vertices T is

$$T = \bigcup_{[b_i, e_i] \in C} \{b_i , e_i\}.$$

The set of vertices T includes all the points of time that serve as delimiters of segments, while every segment in C is also considered as an arc in the graph. In the computation that follows we use the notation that was developed in Section 4 for probabilistic grammars.

The maximal derivation algorithm uses a list of states at every node of the graph (T, C). Each state in that list is a pair $\langle X, b \rangle$, $X \in B$, $b \in I$ (as in Section 4, N is the set of nonterminals and I is a set of indices to the productions). Such a state is used to indicate (as in Section 4) that production b of the grammar was used, and that the production generated the nonterminal X. By finding a maximal likelihood measure for every possible state at every node, the maximum likelihood at a terminal state of T_E will designate the maximal likelihood that any derivation can have. Tracing back from that maximal terminal state will generate a maximal derivation.

Given the graph (T, C), let $T_B = v_0 , v_1 ,..., v_n = T_E$ be an ordering of the vertices in T such that $v_i > v_{i-1}$ for $1 < i \leqslant n$. Let $P_{v_i}(\langle A, b \rangle)$ indicate the

merit value assigned to the state $\langle A, b\rangle$ at node v_i. We begin the computation of merit values by the initial assignment of all $P_{v_0}(\langle N, i\rangle) = 0$, except for $P_{v_0}(\langle S, 0\rangle) = 1$. This assignment indicates that initially only productions having S (which is the starting symbol of G) on the left-hand side can be used, and according to the initial probability vector φ_0 (see Section 2 for the definition of probabilistic grammar). Then, for every $v_i \in T$, $i = 1,..., n$, and for every state $\langle N, j\rangle$, the following is computed:

$$P_{v_i}(\langle N, j\rangle) = \operatorname*{Max}_{\langle M,k\rangle} \{ \operatorname*{Max}_{e=[v,v_i]\in C} \{P_v(\langle M, k\rangle) \cdot \varphi_k(j) \cdot P_e(t_j)\}\}, \tag{9}$$

where $e = [v, v_i] \in C$ is a segment starting at v and terminating at v_i, $\langle M, k\rangle$ is a state ranging over all possible states at v, and production j of G is $M \to t_j N$. Expression (9) is explained as follows: If the merit at state $\langle M, k\rangle$ of node v is $P_v(\langle M, k\rangle)$, and production j is used (where production j is $[j\colon M \to t_j N; \varphi_j]$) to get to node v_i using the arc $e = [v, v_j]$, then we arrive at the state $\langle N, j\rangle$ of v_i, with merit $P_v(\langle M, k\rangle) \cdot \varphi_k(j) \cdot P^{\alpha}(t_j)$. This merit is the product of three elements:

(1) $P_v(\langle M, k\rangle)$, the merit at node v when we have the nonterminal M and the last production used was k;

(2) $\varphi_k(j)$, the probability in the grammar of using production j after production k was used; and

(3) $P_e(t_j)$, the probability that t_j, the terminal produced by production j, is the interpretation of the segment $e = [v, v_j]$ as given by the probability vector assigned to this segment.

Computing (9) assures that every element at each node will get the maximal possible merit out of all the possible ways to reach it. It can be shown that the initial assignments of merit values for P_{v_0}, together with (9), ensure that the maximum likelihood at a terminal state of T_E is also the maximal value for (8) when rewriting (8) as was done in (4).

In order to be able to trace the maximal derivation back from the terminal state having maximal merit at T_E, pointers are needed from every state to a preceding state that maximized its merit. This can be accomplished by the following definition.

Let $F_{v_i}(\langle N, j\rangle)$ be a triple $\langle v, M, j\rangle$ such that the maximal merit $P_{v_i}(\langle N, j\rangle)$ was maximized from the state $\langle M, j\rangle$ of the node v. Then, having the F's, we can trace back a derivation from the maximal element of T_E. Let $(\langle v_k, X_k, i_k\rangle)_{k=1}^{L}$ be a sequence of elements such that $v_0 = T_B$, $X_0 = S$, $\langle v_{k-1}, X_{k-1}, i_{k-1}\rangle = F_{v_k}(\langle X_k, i_k\rangle)$, $v_L = T_E$, and $\langle X_L, i_L\rangle$ is the terminal state having maximal merit in T_E. Then $i_1,..., i_L$ from that sequence is a maximal derivation for the complete probabilistic configuration PC and the grammar G_p, producing as the maximal interpretation the string $t_{i_1},..., t_{i_L}$.

8. Concluding Remarks

The algorithms in this paper find the maximal derivation, since every possible derivation corresponds to some path in the structure represented by the P's and F's. All the nonmaximal subpaths are disregarded since every subpath of a maximal path is itself maximal.

The complexity of computing the maximal probabilities for a new position is independent of the length of the input, but the number of positions is dependent on the length of the input. For a finite-state grammar and a string of length L (or an ambiguous segmentation with L segments) we have $L + 1$ possible positions. For a linear grammar, the positions for an input of length L can be arranged in $L + 1$ levels as in Section 5, where at the kth level there are k positions. The total number of positions is $1 + 2 + \cdots + (L + 1) = \frac{1}{2}(L + 1)(L + 2)$, so that the complexity is proportional to the square of L.

The algorithms described in this paper can be used to find maximal derivations for probabilistic strings that are derived from a stochastic language modeled by a finite-state or linear probabilistic grammar. Using a maximal derivation as an unambiguous interpretation for a probabilistic string is different from using an error-correcting algorithm (Neuhoff, 1975, Kashyap and Mittal, 1977). Error-correcting methods try to change a classification that has already been made, while a maximal derivation will only choose one interpretation from those that have nonzero value in eaeh probability vector. Another algorithm that selects one interpretation from a given set of probability vectors is relaxation labeling (Rosenfeld *et al.*, 1976; Zucker, 1976; Peleg, 1979). But since relaxation uses only limited statistical knowledge about the source of the language, the optimality of its solution has not been established. For the class of cases that can be modeled by simple probabilistic grammars, maximal derivations are more attractive than relaxation, considering their optimality and low time complexity as discussed in this paper.

When looking only for a maximal interpretation where a derivation is of no interest, it is reasonable to consider for a given string x the sum of the probabilities of all its derivations, rather than the probability of one (maximal) derivation. But it is much harder to sum the probabilities over all derivations; this problem is beyond the scope of this paper.

Received: September 15, 1978

References

Aho, A. V., and Ullman, L. D. (1972), "The Theory of Parsing, Translation, and Compiling," Vol. 1: "Parsing," Prentice–Hall, Englewood Cliffs, N. J.

Albus, J. E. (1977), Electrocardiogram interpretation using a stochastic finite state model, *in* "Syntactic Pattern Recognition Applications" (K. S. Fu, Ed.), pp. 51–64, Springer–Verlag, Berlin/New York.

FORNEY, G. D. (1973), The Viterbi algorithm, *Proc. IEEE* **61**, 268–278.

FU, K. S. (1974), "Syntactic Methods in Pattern Recognition," Academic Press, New York/London.

KASHYAP, R. L., AND MITTAL, M. C. (1977), A new method for error correction in strings with applications to spoken word recognition, *in* "Proceedings, IEEE Conference on Pattern Recognition and Image Processing," pp. 76–82.

NEUHOFF, D. L. (1975), The Viterbi algorithm as an aid in text recognition, *IEEE Trans. Information Theory* **IT-21**, 222–226.

NILSSON, N. (1971), "Problem Solving Methods in Artificial Intelligence," McGraw–Hill New York.

PELEG, S. (1979), "Ambiguity Reduction in Handwriting with Ambiguous Segmentation and Uncertain Interpretation," *Computer Graphics Image Processing* **10**, 235–245.

ROSENFELD, A., ZUCKER, S., AND HUMMEL, R. (1976), Scene labeling by relaxation operations, *IEEE Trans. Systems Man Cybernet.* **SMC-6**, 420–433.

SALOMAA, A. (1969), Probabilistic and weighted grammars, *Inform. Contr.* **15**, 529–544.

VELASCO, F. R. D., AND ROSENFELD, A. (1978), "The Application of Relaxation to Waveforms with Ambiguous Segmentation," *IEEE Trans. Systems Man Cybernet.*, in press.

ZUCKER, S. (1976), Relaxation labeling and the reduction of local ambiguities, *in* "Proceedings, Third International Joint Conference on Pattern Recognition," pp. 852–861.